| **CS787**: Advanced Algorithms | **Homework 1** |
|---|---|
| **Out:** Jan 31, 2022 | **Due:** 2/10/22 |

# 1 Hashing Bashing

Two-level hashing is nice because, for $n$ items, it can achieve perfect hashing with just $O(n)$ buckets. However, it does have the drawback that the perfect hash function has a lengthy description (since you have to describe the second-level hash function for each bucket).

(a) For a universe $U$ of size $|U|$, how many bits are required to describe a perfect two-level hash function that is implemented using 2-universal hash functions? We're just looking for big-O notation.

Consider the following alternative approach to producing a perfect hash function with a small description. Define bi-bucket hashing, or bashing, as follows. Given $n$ items, allocate two arrays of size $O(n^{1.5})$. When inserting an item, map it to one bucket in each array, and place it in the emptier of the two buckets.

(b) Suppose a fully random function is used to map each item to buckets. Prove that the expected number of collisions is $O(1)$.
**Hint:** What is the probability that the $k$-th inserted item collides with some previously inserted item?

(c) Show that bashing can be implemented efficiently, with the same expected outcome, using random hash functions from 2-universal hashing. How large is the description of the resulting function?

(d) Conclude an algorithm that requires just O(n) hash function evaluations in expectation to identify a perfect bash function for a set of n items.

(e) (Optional) Generalize the above approach to use less space by exploiting tri-bucket hashing (trashing), quad-bucket hashing (quashing), and so on.

# 2 Lower bound for Balls and Bins

In class we showed that $n$ balls in $n$ random bins see a max load of $O(\log n/\log\log n)$. Show this bound is tight.

(a) Show there is a $k = \Omega(\log n/\log\log n)$ such that bin 1 has $k$ balls with probability at least $1/\sqrt{n}$. An inequality that might be helpful:

$$\left(\frac{n}{k}\right)^k \le \binom{n}{k} \le \left(\frac{en}{k}\right)^k$$

(b) Prove that conditioning on the first bin not having $k$ balls only increases the probability that the second bin does, and so on. Conclude that with high probability, some bin has $\Omega(\log n/\log\log n)$ balls. When we say "high probability", we mean with probability at least $1 - 1/n$, although your bound could be much higher.

# 3 Rank estimation from a sample

Suppose we have an unsorted list of distinct numbers $x_1, x_2, ..., x_n$. We want to randomly subsample $t < n$ items from this list (drawing with replacement) and from that subsample we want to return some element $x$ such that rank$(x)$ is approximately equal to $k$ for a given $k$. By rank we mean the rank of $x$ in the original list. The largest of $x_1, ..., x_n$ has rank 1, the second largest has rank 2, etc. and we're trying to find something with rank $\approx k$.

Describe a simple strategy for choosing a candidate $x$ from the subsample with rank approximately equal to $k$. How large do we need to set $t$ (in big-O notation) so that, with probability $(1 - \delta)$, your strategy returns an $x$ with $(1 - \epsilon)k \le \text{rank}(x) \le (1 + \epsilon)k$?
**Hint:** This should not require any complex counting arguments or combination/permutation calculations.

# 4 Counting and Sampling through Queries

Your friend has a set of numbers $S \subseteq \{1, ..., n\}$ in his mind. If you pick a set $Q \subseteq \{1, ..., n\}$, and present it to your friend he will answer if the intersection $S \cap Q$ is empty or not. Your goal is to figure out (approximately) how many elements are contained in the set $S$.

(a) Devise a strategy that distinguishes whether $S$ contains $\leq k$ or $\geq (1+\epsilon)k$ for any $k \in \{1, ..., n\}$. Show that $O(\frac{\log(1/\delta)}{\epsilon^2})$ queries are sufficient to distinguish the two cases with probability at least $1-\delta$.
**Hint:** Consider picking a random set $Q$ that contains every element $i \in \{1, ..., n\}$ with probability $1/k$.

(b) Show that this implies an efficient estimator $\hat{s}$ for $|S|$, that with probability at least $2/3$ satisfies $\hat{s} \leq |S| \leq (1+\epsilon)\hat{s}$ using the aforementioned strategy. The number of queries should be at most logarithmic in $n$.

(c) Assuming you know the size $|S|$, devise an efficient scheme that samples a uniformly random element from $S$ by querying your friend multiple times with different sets $Q$. The (expected) number of queries to produce a single number should be at most logarithmic in $n$.
**Hint:** Can you find a random set $Q$ that contains **exactly one** element from $S$? Can you think of an efficient method to check whether $Q$ satisfies this property, and if it does, return the unique element in the intersection?